НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

(повна назва інституту/факультету)

КАФЕДРА інформатики та програмної інженерії

(повна назва кафедри)

**КУРСОВА РОБОТА**

з дисципліни «Бази даних»

(назва дисципліни)

на тему:\_База даних для підтримки діяльності call-центру\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студента (ки) \_\_2\_\_ курсу \_ІП-24\_\_\_\_\_ групи

спеціальності 121 «Інженерія програмного забезпечення»

\_\_\_\_\_\_\_\_\_\_\_\_\_Піддубного Б.С. \_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник \_Ліщук Катерина Ігорівна, доц., к.т.н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_ Оцінка ECTS \_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ – 2024 рік

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет Інформатики та обчислювальної техніки

(повна назва)

Кафедра Інформатики та програмної інженерії

(повна назва)

Дисципліна Бази даних

Курс \_\_2\_\_\_ Група \_\_ІП-24\_\_ Семестр \_\_\_3\_\_

**З А В Д А Н Н Я**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Піддубному Борису Сергійовичу\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1. Тема роботи База даних для підтримки діяльності call-центру\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

керівник роботи \_\_Ліщук Катерина Ігорівна, к.т.н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи \_\_\_\_\_29.12.2023\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

завдання на розробку бази даних для підтримки діяльності call-центру \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1) Аналіз предметного середовища

2) Побудова ER-моделі

3) Побудова реляційної схеми з ER-моделі

4) Створення бази даних, у форматі обраної системи управління базою даних

5) Створення користувачів бази даних

6) Імпорт даних з використанням засобів СУБД в створену базу даних

7) Створення мовою SQL запитів

8) Оптимізація роботи запитів

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Дата видачі завдання\_\_\_\_04.11.2023**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсового проекту | Строк виконання етапів проекту | Примітка |
| 1 | Аналіз предметного середовища | 12.11.2023 |  |
| 2 | Побудова ER-моделі | 12.12.2023 |  |
| 3 | Побудова реляційної схеми з ER-моделі | 13.12.2023 |  |
| 4 | Створення бази даних, у форматі обраної системи управління базою даних | 17.12.2023 |  |
| 5 | Створення користувачів бази даних | 19.12.2023 |  |
| 6 | Імпорт даних з використанням засобів СУБД в створену базу даних | 20.12.2023 |  |
| 7 | Створення мовою SQL запитів | 24.12.2023 |  |
| 8 | Оптимізація роботи запитів | 07.01.2023 |  |
| 9 | Оформлення пояснювальної записки | 20.01.2023 |  |
| 10 | Захист курсової роботи | 24.01.2023 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Студент \_\_\_\_\_\_\_\_\_\_\_\_** *Піддубний Б.С*

(підпис ) (прізвище та ініціали)

**Керівник роботи *\_\_\_\_\_\_\_\_\_\_\_\_\_*** *Ліщук К.І*

(підпис ) (прізвище та ініціали)

Анотація

Пояснювальна записка до курсової роботи: 67 сторінки, 52 рисунків, 12 таблиць, 7 посилань.

Об’єкт дослідження: база даних для підтримки діяльності call-центру.

Мета роботи: розробка та реалізація бази даних для підтримки діяльності call-центру.

Проведено аналіз предметного середовища, побудована ER-модель та реляційна схема, створена база даних у форматі MySQL. Створені користувачі та ролі. Були імпортовані дані в створену базу даних. Створені функції, процедури, тригери, представлення та SQL запити для роботи з базою даних. Також була проведена оптимізація роботи запитів.

Виконана реалізація бази даних для підтримки діяльності call-центру у форматі MySQL згідно з варіантом завдання.

Зміст

[**Вступ** 5](#_Toc156798936)

[**1** **Аналіз предметного середовища** 6](#_Toc156798937)

[1.1 Опис предметного середовища 6](#_Toc156798938)

[1.2 Аналіз існуючих програмних продуктів 6](#_Toc156798939)

[**2** **Постановка завдання** 9](#_Toc156798940)

[**3** **Побудова ER-моделі** 10](#_Toc156798941)

[3.1 Бізнес-правила 10](#_Toc156798942)

[3.2 Створення сутностей 10](#_Toc156798943)

[3.3 Опис сутностей 11](#_Toc156798944)

[**4** **РЕАляційна модель бази даних** 15](#_Toc156798945)

[**5** **реалізація бази даних** 20](#_Toc156798946)

[5.1 Створення бази даних 20](#_Toc156798947)

[5.2 Імпортування даних 23](#_Toc156798948)

[**6** **Створення користувачів бази даних** 28](#_Toc156798949)

[6.1 Адміністратор call-центру 28](#_Toc156798950)

[6.2 Наглядач call-центру 28](#_Toc156798951)

[6.3 Агент call-центру 28](#_Toc156798952)

[6.4 Користувач call-центру 29](#_Toc156798953)

[**7** **робота з базою даних** 30](#_Toc156798954)

[7.1 Тексти генераторів 30](#_Toc156798955)

[7.2 Тексти збережених процедур/функцій. 31](#_Toc156798956)

[7.3 Тригери 42](#_Toc156798957)

[7.4 Представлення 48](#_Toc156798958)

[7.5 SQL-запити 51](#_Toc156798959)

[7.6 Індекси та результати оптимізації 62](#_Toc156798960)

[**Висновки** 66](#_Toc156798961)

[**Список використаної літератури** 67](#_Toc156798962)

**Вступ**

У сучасному світі, важливість ефективного управління та обробки даних надзвичайно висока. Однією з ключових складових цього процесу є використання баз даних, які забезпечують систематизоване зберігання та зручний доступ до інформації.

У даній курсовій роботі розробляється база даних для підтримки діяльності call-центру, на прикладі компанії з надання технічної підтримки та IT послуг, з використанням системи управління базами даних MySQL[1]. Діяльність call-центрів на пряму пов’язана із взаємодією з величезними обсягами даних про клієнтів, статистикою, журналами викликів тощо. Призначенням розробки є поліпшення ефективності роботи call-центру, скорочення часових затрат на обробку інформації, а також забезпечення надійності та цілісності даних. Сфера використання розробки охоплює всі аспекти, що пов'язані із обслуговування клієнтів, наданням послуг та управління великим обсягом інформації.

Вибір MySQL для реалізації бази даних обґрунтований його широким застосуванням, надійністю та ефективністю. Також важливим аспектом у виборі MySQL відіграє його відкритість та вільна ліцензія, що забезпечує доступність для різних користувачів та розробників.

Отже, дана курсова робота націлена на створення бази даних для оптимізації діяльності call-центру, яка відповідатиме сучасним вимогам та сприятиме підвищенню ефективності обслуговування клієнтів.

**1 Аналіз предметного середовища**

1.1 Опис предметного середовища

База даних call-центру, повинна включати в себе велику кількість інформації про: клієнтів, робітників, дзвінків, наявних послуг, відгуків про роботу агентів call-центру тощо.

Вхідними даними є інформація про: клієнтів (контактна інформація, історія дзвінків на гарячу лінію, відгуки), робітників call-центру (контактна інформація, зарплата, дата найму), також індекси для наглядачів та агентів, дзвінки (час початку, час закінчення, інформація про клієнта та агента call-центру), предмет дискусії (опис), послуги (назва, ціна) та відгуки (рейтинг, коментарі)

Вихідними даними виступають: звіти про продуктивність агентів від наглядачів. У звітах вказана статистика дзвінків, аналіз надання послуг, відгуки клієнтів тощо.

1.2 Аналіз існуючих програмних продуктів

Five9[2] – це сучасна хмарна платформа для call-центрів, яка надає повний спектр можливостей для ефективної комунікації з клієнтами. Five9 пропонує обробку дзвінків та чатів, електронну пошту, соціальні мережі та інші канали в одному інтерфейсі. Можна легко переключатися між різними каналами, а також використовувати функції, такі як переведення дзвінків, конференц-зв’язок, утримання дзвінків, запис голосових повідомлень, тощо

Наявна інтелектуальна маршрутизація дзвінків, автоматизація вихідних дзвінків, CRM-інтеграція (Управління відносинами з клієнтами), аналітика та звітність. Також, Five9 підтримує більше 100 мов та діалектів і використовує штучний інтелект для покращення роботи агентів та задоволення клієнтів. Наприклад, Five9 може автоматично маршрутизувати дзвінки до найбільш підходящих агентів, а також надавати їм підказки та рекомендації під час розмови. Використовується IVA (інтелектуальний віртуальний асистент).

На відміну від Five9, що є хмарною платформою, моя база даних може бути розміщена на власному сервері, що надає більший контроль над безпекою та доступом до даних.

Наступна програма – VoIPTime[3]. Вона створена для call-центрів та пропонує найкраще співвідношення ціна-якість. VoIPTime також дозволяє обробляти дзвінки, чати, електронну пошту, соціальні мережі та інші канали в одному інтерфейсі. Серед наявних функції є:

* Автоматичне нагадування
* Запис дзвінків
* Сценарії розмов (дозволяє створювати та використовувати готові сценарії для різних типів дзвінків, таких як привітання, продажі, консультації, підтримка)
* Аналітика та звітність (схожа за принципом дії на таблицю з відгуками про агентів call-центру в моїй базі даних).

Також, VoIPTime надає функції використання IVR (інтерактивна голосова відповідь) та чатботів для самообслуговування клієнтів, що суттєво зменшує навантаження на агентів та підвищує задоволення клієнтів.

В сучасному світі, де бізнес залежить від задоволення клієнтів, важливо мати ефективні та надійні інструменти для аналітики та звітності, які допомагають краще зрозуміти бізнес та взаємодію з клієнтами. Обидва продукти, Five9 та VoIPTime, надають такі інструменти, які дозволяють отримувати цінну інформацію про продуктивність, якість, задоволеність та поведінку клієнтів. Використання наявних функцій допоможе забезпечити високу якість обслуговування, зменшити витрати та відрізнитися від конкурентів.

Однак, не всі продукти та програмне забезпечення однаково підходять для різних call-центрів. У підсумку можна зазначити, що вибір конкретного продукту та програмного забезпечення повинен ґрунтуватися на відповідних потребах call-центру, його розміру, специфіки та цілей. Повинні враховуватися питання швидкість обробки даних, можливості реструктуризації, збільшення бази даних. Також значну роль відіграє актуальність ПЗ у сучасних реаліях, де технології швидко розвиваються та впливають на очікування клієнтів. Call-центр може використовувати хмарні рішення, які дозволяють легко масштабувати, інтегрувати та оновлювати свої системи, або застосовувати штучний інтелект, який допомагає автоматизувати та оптимізувати процеси.

Тому, на підгрунті цих фактів виникає потреба створення власної бази даних для підтримки діяльності call-центру із власними встановленими правилами та можливостями. Така база даних дозволить call-центру керувати своїми даними, адаптуватися до змін, використовувати свої ресурси ефективно та забезпечити високий рівень задоволення клієнтів. Розробка та впровадження такої бази даних стане важливим кроком у напрямку оптимізації та автоматизації процесів у сфері обслуговування клієнтів, сприяючи підвищенню ефективності та якості роботи call-центрів.

**2 Постановка завдання**

Метою даної курсової роботи є розробка та реалізація бази даних для call-центра компанії з надання технічної підтримки та IT послуг. Основними напрямками роботи є поліпшення ефективності роботи call-центру, скорочення часових затрат на обробку інформації, а також забезпечення надійності та цілісності даних .

Етапи розробки курсової роботи:

1. Аналіз предметного середовища
2. побудова ER-моделі
3. Побудова реляційної схеми бази даних
4. Створення бази даних з використання СУБД MySQL
5. Імпорт даних з використанням засобів СУБД
6. Автоматизація ключових бізнес-процесів за допомогою запитів мовою SQL
7. Оптимізація роботи запитів
8. Додання користувачів у систему

База даних повинна відповідати стандартам ефективного управління роботи в кол-центрі, бути стійкою до навантажень, мати можливість впровадити нові функції та/або сутності, забезпечити швидкий доступ до інформації. Дані в базі повинні зберігатися відповідно до політики конфіденційності компанії.

Серед присутніх таблиць у базі даних слід визначити наступні сутності:

* Користувач – людина, що звернулася до кол-центру з метою отримання послуг, консультації або вирішення проблеми.
* Агент – людина та працівник компанії, що проводить консультативні розмови з користувачом, пропонує послуги.
* Наглядач (супервайзер) – людина та працівник компанії, що наймає в свою команду агентів, слідкує за їхньої успішностю.

**3 Побудова ER-моделі**

3.1 Бізнес-правила

До основних бізнес-правил належить:

1. Наглядач може не мати підпорядкованих працівників на даний момент часу
2. Кожному працівнику та користувачу відповідають унікальні контактні дані
3. Користувач зареєстрован у базі під конкретними даними, для їх зміни потрібно звертатися до кол-центру
4. Номер телефону повинен бути унікальним
5. Пошта повинна бути унікальною
6. Зарплата робітників компанії не може бути від’ємною або більше за 99999.99 доларів
7. Користувач може залишити відгук про агента та оцінити його від 1 до 5
8. Дзвінок не може тривати більше 2 годин
9. Ціна послуг не може бути від’ємною або більше за 9999.99 доларів
   1. Створення сутностей

Для виконання поставленої задачі було створено наступні стуності:

1. Contact\_info
2. Customer
3. Agent
4. Supervisor
5. Staff
6. Issue
7. Feedback
8. Call
9. Service

3.3 Опис сутностей

Після детального аналізу опису предметного можна точно виділити наступні атрибути сутностей (таблиця 3.1).

Таблиця 3.1 – Опис сутностей

|  |  |  |
| --- | --- | --- |
| Назва сутності | Атрибути | Опис |
| contact\_info | id | Унікальний ідентифікатор контактної інформації |
| name | Повне ім’я людини |
| number | Номер телефона |
| email | Електрона пошта |
| customer | id | Унікальний ідентифікатор користувача |
| contact\_info\_id | Ідентифікатор контактної інформації |
| agent | id | Унікальний ідентифікатор агента |
| staff\_id | Ідентифікатор працівника |
| supervisor\_id | Ідентифікатор наглядача |
| supervisor | id | Унікальний ідентифікатор наглядача |
| staff\_id | Ідентифікатор працівника |

Продовження таблиці 3.1

|  |  |  |
| --- | --- | --- |
| Назва сутності | Атрибути | Опис |
| staff | id | Унікальний ідентифікатор працівника |
| salary | Зарплата працівника |
| hire\_date | Дата найму |
| contact\_info\_id | Ідентифікатор контактної інформації |
| issue | id | Унікальний ідентифікатор предмету дискусії |
| description | Опис |
| feedback | id | Унікальний ідентифікатор відгука |
| rating | Оцінка роботи агента |
| comments | Коментар до відгуку |
| customer\_id | Ідентифікатор користувача |
| agent\_id | Ідентифікатор агента |
| call | id | Унікальний ідентифікатор дзінка |
| start\_at | Початок дзвінка |
| end\_at | Кінець дзвінка |
| customer\_id | Ідентифікатор користувача |
| agent\_id | Ідентифікатор агента |
| service | id | Унікальний ідентифікатор послуги |
| name | Назва |
| price | Ціна |

Опис зв’язків між сутностями:

1. contact\_info – customer: один до одного (1:1). Для кожного користувача існує відповідна контактна інформація
2. contact\_info – staff: один до одного (1:1). Для кожного працівника існує відповідна контактна інформація.
3. staff – agent: один до одного (1:1). Для кожного агента існує відповідний запис працівника в компанії.
4. staff – supervisor: один до одного (1:1). Для кожного наглядача існує відповідний запис працівника в компанії.
5. customer – feedback: один до багатьох (1:N). Користувач може залишити багато відгуків.
6. agent – feedback: один до багатьох (1:N). Агент може мати багато відгуків.
7. customer – call: один до багатьох (1:N). Користувач може здійснити багато дзвінків.
8. agent – call: один до багатьох (1:N). Агент може відпрацювати багато дзвінків.
9. call – service: багато до багатьох (M:N). Під час дзвінка можна замовити багато послуг. Послугу можуть замовити в багатьох дзвінках. Зв’язок виконан з використання асоціативної таблиці.
10. call – issue: багато до багатьох (M:N). Під час дзвінка можна порушити багато питань та провести багато дискусій. Предмет дискусії та порушені питання можуть обговорюватися в багатьох дзвінках.

Побудована ER-модель зображена на рис. 3.1. Кожна таблиця має свої ключові атрибути та логічні зв’язки з іншими таблицями. Для зв’язків з іншими сутностями використовуються FK (зовнішні ключі).



Рис. 3.1 – ER-модель бази даних для підтримки діяльності call-центру

**4 РЕАляційна модель бази даних**

Для ефективної роботи call-центру необхідно мати базу даних, яка зберігає, обробляє і надає доступ до великої кількості інформації про клієнтів, співробітників, статистику тощо. Саме для такої цілі було обрано систему управління базами даних (СУБД) MySQL1. MySQL – Це популярна, відкрита і надійна СУБД, яка має багато переваг для цієї мети. Ось деякі з них:

* Open source[4]. MySQL є однією з найпоширеніших та високопродуктивних реляційних СУБД, а відкритий вихідний код дозволяє безкоштовно використовувати та модифікувати систему відповідно до конкретних потреб проєкту.
* Швидкість і продуктивність. MySQL володіє високою швидкістю обробки запитів і може ефективно працювати з великими обсягами даних. Це дозволяє call-центру забезпечити швидку і якісну обслуговування клієнтів, а також аналізувати і оптимізувати свою діяльність.
* Безпека і надійність. MySQL має різні рівні безпеки, які захищають дані від несанкціонованого доступу, змін або втрат. Він дозволяє встановлювати паролі, обмеження та інші заходи, які гарантують конфіденційність і цілісність даних. MySQL також підтримує резервне копіювання і відновлення даних, що знижує ризик втрати важливої інформації.

Після детального обгрунтування вибору СУБД було реалізовано та описано структури бази даних у табличному вигляді. Таблиці 4.1 – 4.11.

Таблиця 4.1 – «contact\_info» містить контактну інформацію про працівників та користувачів call-центру. Структура таблиці наступна:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** |
| id | INT |  | PK | Унікальний ідентифікатор контактної інформації |
| name | VARCHAR | 100 |  | Повне ім’я людини |
| number | VARCHAR | 15 |  | Номер телефону |
| email | VARCHAR | 100 |  | Електрона пошта |

Таблиця 4.2 – «customer» містить інформацію про користувачі call-центру. Структура таблиці наступна:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** |
| id | INT |  | PK | Унікальний ідентифікатор користувача |
| contact\_info\_id | INT |  | FK | Ідентифікатор контактної інформації |

Таблиця 4.3 – «agent» містить інформацію про агентів call-центру. Структура таблиці наступна:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** |
| id | INT |  | PK | Унікальний ідентифікатор агента |
| staff\_id | INT |  | FK | Ідентифікатор працівника |
| supervisor\_id | INT |  | FK | Ідентифікатор наглядача |

Таблиця 4.4 – «supervisor» містить інформацію про наглядачів call-центру. Структура таблиці наступна:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** |
| id | INT |  | PK | Унікальгий ідентифікатор наглядача |
| staff\_id | INT |  | FK | Ідентифікатор працівника |

Таблиця 4.5 – «staff» містить інформацію про працівників call-центру. Структура таблиці наступна:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** |
| id | INT |  | PK | Унікальний ідентифікатор працівника |
| salary | DECIMAL | 7, 2 |  | Заробітна плата |
| hire\_date | DATE |  |  | Дата найму на роботу |
| contact\_info\_id | INT |  | FK | Ідентифікатор контактної інформації |

Таблиця 4.6 – «feedback» містить інформацію про відгуки користувачі. Структура таблиці наступна:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** |
| id | INT |  | PK | Унікальний ідентифікатор відгука |
| rating | ENUM |  |  | Оцінка роботи агента |
| comments | TEXT | 1000 |  | Коментар до відгуку |
| customer\_id | INT |  | FK | Ідентифікатор користувача |
| agent\_id | INT |  | FK | Ідентифікатор агента |

Таблиця 4.7 – «issue» містить інформацію про теми проведених дискусій, можливі порушені питання під час дзвінків, предмети дискусії. Структура таблиці наступна:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** |
| id | INT |  | PK | Унікальний ідентифікатор предмету дискусії |
| description | VARCHAR | 255 |  | Опис |

Таблиця 4.8 – «service» містить інформацію про послуги, що може надати компанії користувачеві. Структура таблиці наступна:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** |
| id | INT |  | PK | Унікальний ідентифікатор послуги |
| name | VARCHAR | 100 |  | Назва |
| price | DECIMAL | 6, 2 |  | Ціна |

Таблиця 4.9 – «call» містить інформацію про дзвінки в call-центрі. Структура таблиці наступна:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** |
| id | INT |  | PK | Унікальний ідентифікатор дзінка |
| start\_at | TIMESTAMP |  |  | Початок дзвінка |
| end\_at | TIMESTAMP |  |  | Кінець дзвінка |
| customer\_id | INT |  | FK | Ідентифікатор користувача |
| agent\_id | INT |  | FK | Ідентифікатор агента |

Таблиця 4.10 – «call\_issue» асоціативна, містить інформацію про зв’язок між дзвінками та предметами дискусії в call-центрі. Структура таблиці наступна:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** |
| call\_id | INT |  | FK | Ідентифікатор дзвінка |
| issue\_id | INT |  | FK | Ідентифікатор предмету дискусії |

Таблиця 4.11 – «call\_service» асоціативна, містить інформацію про зв’язок між дзвінками та послугами в call-центрі. Структура таблиці наступна:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** |
| call\_id | INT |  | FK | Ідентифікатор дзвінка |
| service\_id | INT |  | FK | Ідентифікатор послуги |

Після наведення опису структури в табличному вигляді була побудована реляційна схема за допомогою СУБД MySQL (рис. 4.1)

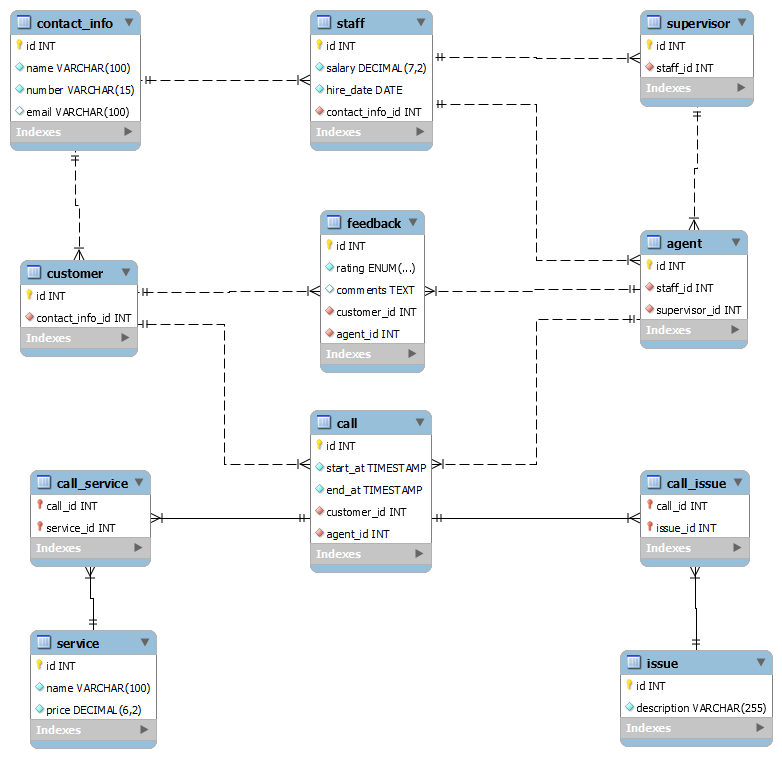


Рис. 4.1 – Реляційна схема бази даних згенерована в додатку MySQL Workbench[5]

Підводемо підсумки розділу: було створено сутності, побудовано логічні зв’язки між ними, та на їх основі розроблено ER-модель бази даних для підтримки діяльності call-центру. Також, було свторено реляційну схему за допомогою MySQL Workbench.

**5 реалізація бази даних**

5.1 Створення бази даних

Результатом проєктування бази даних є сформований SQL-скрипт, який використовується для створення таблиць, які були наведені в ER-моделі та в табличних описах:

CREATE TABLE `contact\_info` (

`id` int PRIMARY KEY AUTO\_INCREMENT,

`name` varchar(100) NOT NULL,

`number` varchar(15) UNIQUE NOT NULL,

`email` varchar(100) UNIQUE

);

CREATE TABLE `customer` (

`id` int PRIMARY KEY AUTO\_INCREMENT,

`contact\_info\_id` int UNIQUE NOT NULL,

FOREIGN KEY (`contact\_info\_id`) REFERENCES `contact\_info` (`id`)

);

CREATE TABLE `staff` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`salary` DECIMAL(7, 2) NOT NULL,

`hire\_date` DATE NOT NULL,

`contact\_info\_id` INT UNIQUE NOT NULL,

CHECK (`salary` > 0),

FOREIGN KEY (`contact\_info\_id`) REFERENCES `contact\_info` (`id`)

);

CREATE TABLE `supervisor` (

`id` int PRIMARY KEY AUTO\_INCREMENT,

`staff\_id` int UNIQUE NOT NULL,

FOREIGN KEY (`staff\_id`) REFERENCES `staff` (`id`)

);

CREATE TABLE `agent` (

`id` int PRIMARY KEY AUTO\_INCREMENT,

`staff\_id` int UNIQUE NOT NULL,

`supervisor\_id` int NOT NULL,

FOREIGN KEY (`staff\_id`) REFERENCES `staff` (`id`),

FOREIGN KEY (`supervisor\_id`) REFERENCES `supervisor` (`id`)

);

CREATE TABLE `feedback` (

`id` int PRIMARY KEY AUTO\_INCREMENT,

`rating` enum('1','2','3','4','5') NOT NULL,

`comments` text(1000),

`customer\_id` int NOT NULL,

`agent\_id` int NOT NULL,

FOREIGN KEY (`customer\_id`) REFERENCES `customer` (`id`),

FOREIGN KEY (`agent\_id`) REFERENCES `agent` (`id`)

);

CREATE TABLE `issue` (

`id` int PRIMARY KEY AUTO\_INCREMENT,

`description` varchar(255) NOT NULL

);

CREATE TABLE `service` (

`id` int PRIMARY KEY AUTO\_INCREMENT,

`name` varchar(100) NOT NULL,

`price` decimal(6, 2) NOT NULL,

CHECK (`price` >=0 )

);

CREATE TABLE `call` (

`id` int PRIMARY KEY AUTO\_INCREMENT,

`start\_at` timestamp NOT NULL,

`end\_at` timestamp NOT NULL DEFAULT current\_timestamp,

`customer\_id` int NOT NULL,

`agent\_id` int NOT NULL,

CHECK (`end\_at` <= `start\_at` AND `start\_at` = `end\_at` <= 7200),

FOREIGN KEY (`customer\_id`) REFERENCES `customer` (`id`),

FOREIGN KEY (`agent\_id`) REFERENCES `agent` (`id`)

);

CREATE TABLE `call\_issue` (

`call\_id` int NOT NULL,

`issue\_id` int NOT NULL,

PRIMARY KEY (`call\_id`, `issue\_id`),

FOREIGN KEY (`call\_id`) REFERENCES `call` (`id`),

FOREIGN KEY (`issue\_id`) REFERENCES `issue` (`id`)

);

CREATE TABLE `call\_service` (

`call\_id` int NOT NULL,

`service\_id` int NOT NULL,

PRIMARY KEY (`call\_id`, `service\_id`),

FOREIGN KEY (`call\_id`) REFERENCES `call` (`id`),

FOREIGN KEY (`service\_id`) REFERENCES `service` (`id`)

);

5.2 Імпортування даних

Для імпортування згенерованих даних, що зберігаютсья в текстових файлах у форматі csv (дані розділені комою), треба використати команду LOAD DATA INFILE та вказати шлях до даних. Дані повинні зберігатися в спеціальній теці MySQL\MySQL Server 8.0\Uploads.

1. Імпорт даних у таблицю contact\_info:

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\contact-info.csv'

INTO TABLE `contact\_info`

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(`id`, `name`, `number`, `email`);

1. Імпорт даних у таблицю customer:

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\customer.csv'

INTO TABLE `customer`

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(`id`, `contact\_info\_id`);

1. Імпорт даних у таблицю staff:

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\staff.csv'

INTO TABLE `staff`

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(`id`, `salary`, `hire\_date`, `contact\_info\_id`);

1. Імпорт даних у таблицю supervisor:

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\supervisor.csv'

INTO TABLE `supervisor`

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(`id`, `staff\_id`);

1. Імпорт даних у таблицю agent:

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\agent.csv'

INTO TABLE `agent`

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(`id`, `staff\_id`, `supervisor\_id`);

1. Імпорт даних у таблицю feedback:

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\feedback.csv'

INTO TABLE `feedback`

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(`id`, `rating`, `comments`, `customer\_id`, `agent\_id`);

1. Імпорт даних у таблицю issue:

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\issue.csv'

INTO TABLE `issue`

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\r\n'

IGNORE 1 ROWS

(`id`, `description`);

1. Імпорт даних у таблицю service:

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\service.csv'

INTO TABLE `service`

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\r\n'

IGNORE 1 ROWS

(`id`, `name`, `price`);

1. Імпорт даних у таблицю call:

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\call.csv'

INTO TABLE `call`

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\r\n'

IGNORE 1 ROWS

(`id`, `start\_at`, `end\_at`, `customer\_id`, `agent\_id`);

1. Імпорт даних у таблицю call\_issue:

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\call-issue.csv'

INTO TABLE `call\_issue`

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\r\n'

IGNORE 1 ROWS

(`call\_id`, `issue\_id`);

1. Імпорт даних у таблицю call-service:

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\call-service.csv'

INTO TABLE `call\_service`

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\r\n'

IGNORE 1 ROWS

(`call\_id`, `service\_id`);

Імпорт даних у таблиця відрізняється через їхній формат запису. Деякі дані були згенеровані штучно, за допомогою застосунків або інтернет додатків (наприклад сайт cobbl[6] та filldb[7]), інша – створена та описана вручну.

**6 Створення користувачів бази даних**

6.1 Адміністратор call-центру

CREATE USER 'callcenter\_administrator'@'localhost' IDENTIFIED BY '12345678';

GRANT ALL PRIVILEGES ON callcenter.\* TO 'callcenter\_administrator'@'localhost';

6.2 Наглядач call-центру

CREATE USER 'callcenter\_supervisor'@'localhost' IDENTIFIED BY '12345678';

GRANT SELECT ON callcenter.feedback TO 'callcenter\_supervisor'@'localhost';

GRANT SELECT, INSERT, DELETE, UPDATE ON callcenter.issue TO 'callcenter\_supervisor'@'localhost';

GRANT SELECT, INSERT, DELETE, UPDATE ON callcenter.service TO 'callcenter\_supervisor'@'localhost';

GRANT SELECT, INSERT, DELETE, UPDATE ON callcenter.agent TO 'callcenter\_supervisor'@'localhost';

GRANT SELECT, INSERT, DELETE, UPDATE ON callcenter.customer TO 'callcenter\_supervisor'@'localhost';

GRANT SELECT, INSERT, DELETE, UPDATE ON callcenter.call\_issue TO 'callcenter\_supervisor'@'localhost';

GRANT SELECT, INSERT, DELETE, UPDATE ON callcenter.call\_service TO 'callcenter\_supervisor'@'localhost';

GRANT SELECT, INSERT, DELETE, UPDATE ON callcenter.`call` TO 'callcenter\_supervisor'@'localhost';

6.3 Агент call-центру

CREATE USER 'callcenter\_agent'@'localhost' IDENTIFIED BY '12345678';

GRANT SELECT ON callcenter.feedback TO 'callcenter\_agent'@'localhost';

GRANT SELECT ON callcenter.issue TO 'callcenter\_agent'@'localhost';

GRANT SELECT ON callcenter.service TO 'callcenter\_agent'@'localhost';

GRANT SELECT ON callcenter.customer TO 'callcenter\_agent'@'localhost';

GRANT SELECT, INSERT ON callcenter.call\_issue TO 'callcenter\_agent'@'localhost';

GRANT SELECT, INSERT ON callcenter.call\_service TO 'callcenter\_agent'@'localhost';

GRANT SELECT, INSERT, UPDATE ON callcenter.`call` TO 'callcenter\_agent'@'localhost';

6.4 Користувач call-центру

CREATE USER 'callcenter\_customer'@'localhost' IDENTIFIED BY '12345678';

GRANT SELECT, INSERT ON callcenter.feedback TO 'callcenter\_customer'@'localhost';

Результат виконання команд для створення користувачів командами SQL на рис. 6.1.

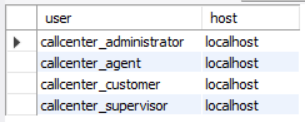


Рис. 6.1 – Створені користувачі в базі даних call-центру

**7 робота з базою даних**

7.1 Тексти генераторів

Було встановлено значення AUTO\_INCREMENT для уніклаьних ідентифікаторів таблиць у базі даних за допомогою наступних SQL команд:

ALTER TABLE contact\_info

MODIFY COLUMN id INT AUTO\_INCREMENT;

ALTER TABLE customer

MODIFY COLUMN id INT AUTO\_INCREMENT;

ALTER TABLE staff

MODIFY COLUMN id INT AUTO\_INCREMENT;

ALTER TABLE supervisor

MODIFY COLUMN id INT AUTO\_INCREMENT;

ALTER TABLE agent

MODIFY COLUMN id INT AUTO\_INCREMENT;

ALTER TABLE feedback

MODIFY COLUMN id INT AUTO\_INCREMENT;

ALTER TABLE issue

MODIFY COLUMN id INT AUTO\_INCREMENT;

ALTER TABLE service

MODIFY COLUMN id INT AUTO\_INCREMENT;

ALTER TABLE call

MODIFY COLUMN id INT AUTO\_INCREMENT;

7.2 Тексти збережених процедур/функцій.

Створено різноманітні процедури та функції для підтримки діяльності call-центру та впровадження функціоналу зазначеного в аналізи предметного середовища

7.2.1 Функція calculate\_last\_month\_earnings

Функція для отримання заробітку call-центру на продажах послуг за останні місяці. Параметром функції виступає кількість місяців. Створена для допомоги керівникам call-центру відслідковувати успішність роботи агентів та call-центру загалом. Результат виконання функції на рис. 7.1.

DELIMITER //

CREATE FUNCTION calculate\_last\_month\_earnings(num\_previous\_months INT) RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

DECLARE total\_price DECIMAL(10,2);

SELECT SUM(service.price) INTO total\_price

FROM callcenter.call\_service

JOIN callcenter.service ON call\_service.service\_id = service.id

JOIN callcenter.`call` ON call\_service.call\_id = `call`.id

WHERE `call`.end\_at >= DATE\_SUB(CURRENT\_DATE, INTERVAL num\_previous\_months MONTH);

RETURN total\_price;

END //

DELIMITER ;

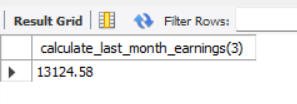


Рис. 7.1 – Результат виконання функції calculate\_last\_month\_earnings для останніх трьох місяців

7.2.2 Процедура agent\_productivity\_report

Дана процедура повертає кількість дзвінків, які обробив агент впродовж зазначеного терміну. Створена для відслідковування продуктивності агенту. Результати виконання процедури на рис. 7.2.

DELIMITER //

CREATE PROCEDURE get\_agent\_total\_calls(IN start\_date DATE, IN end\_date DATE)

BEGIN

SELECT

`call`.agent\_id,

ci.`name` AS agent\_name,

COUNT(\*) AS total\_calls

FROM

callcenter.`call`

JOIN

(

SELECT

agent.id AS agent\_id,

staff.contact\_info\_id

FROM

agent

JOIN

staff ON agent.staff\_id = staff.id

) a ON `call`.agent\_id = a.agent\_id

JOIN

contact\_info ci ON a.contact\_info\_id = ci.id

WHERE

`call`.start\_at >= start\_date AND `call`.end\_at <= end\_date

GROUP BY

a.agent\_id, ci.`name`

ORDER BY

total\_calls;

END //

DELIMITER ;

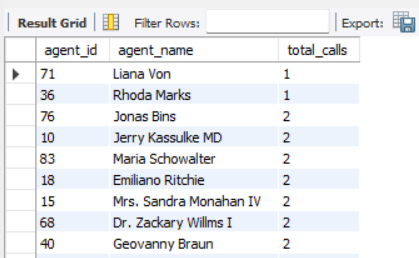


Рис. 7.2 – Результат виконання процедури get\_agent\_total\_calls. Оброблені дзвінки впродовж 2020-01-01 – 2023-12-31

7.2.3 Процедура get\_customer\_feedback

Процедура створена для отримання відгуків від користувачів про кожного агента. Процедура дозволяє дізнатися середній рейтинг агента та загальну кількість відгуків. Результат виконання процедури на рис. 7.3.

DELIMITER //

CREATE PROCEDURE get\_customer\_feedback()

BEGIN

SELECT

agent.id AS agent\_id,

contact\_info.`name` AS agent\_name,

FORMAT(AVG(feedback.rating), 1) AS average\_rating,

COUNT(\*) AS total\_feedbacks

FROM

callcenter.agent

LEFT JOIN callcenter.feedback ON feedback.agent\_id = agent.id

JOIN callcenter.staff ON agent.staff\_id = staff.id

JOIN callcenter.contact\_info ON staff.contact\_info\_id = contact\_info.id

GROUP BY

agent.id;

END //

DELIMITER ;

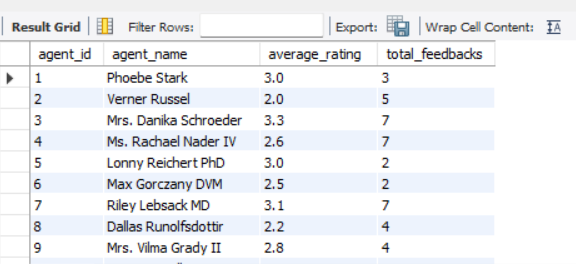


Рис. 7.3 – Результат виконання процедури get\_customer\_feedback

7.2.4 Процедура most\_popular\_services

Процедура відображає кількість дзвінків у яких була замовлена та чи інша послуга. Створена для відслідковування популярності процедур серед клієнтів call-центру. Результат виконання процедури на рис. 7.4

DELIMITER //

CREATE PROCEDURE most\_popular\_services()

BEGIN

SELECT service.id, service.`name`, COUNT(call\_service.call\_id) AS call\_count

FROM service

JOIN call\_service ON service.id = call\_service.service\_id

GROUP BY service.`name`

ORDER BY call\_count DESC;

END //

DELIMITER ;

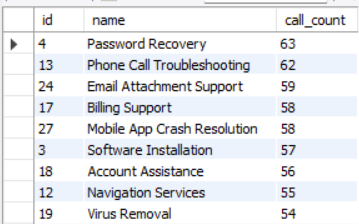


Рис. 7.4 – результат виконання процедури most\_populat\_services

7.2.5 Процедура fetch\_agent\_related\_data

Процедруа створена для відображення всієї доступної інформації про конкретного агента. Використовується для відслідковування даних агента, для їх корегування тощо. Результат виконання процедури для користувача під номером один на рис. 7.5, рис. 7.6 та рис. 7.7.

DROP PROCEDURE IF EXISTS fetch\_agent\_related\_data;

DELIMITER //

CREATE PROCEDURE fetch\_agent\_related\_data(\_agent\_id INT)

BEGIN

SELECT `call`.id AS call\_id, `call`.start\_at, `call`.end\_at, customer\_id, call\_service.service\_id, call\_issue.issue\_id

FROM `call`

LEFT JOIN call\_service ON `call`.id = call\_service.call\_id

LEFT JOIN call\_issue ON `call`.id = call\_issue.call\_id

WHERE `call`.agent\_id = \_agent\_id;

SELECT id AS feedback\_id, rating, comments, customer\_id

FROM feedback WHERE agent\_id = \_agent\_id;

SELECT staff.id AS staff\_id, contact\_info\_id, supervisor\_id, `name`,`number`,email, salary, hire\_date

FROM agent

JOIN staff ON agent.staff\_id = staff.id

JOIN contact\_info ON staff.contact\_info\_id = contact\_info.id

WHERE agent.id = \_agent\_id;

END //

DELIMITER ;

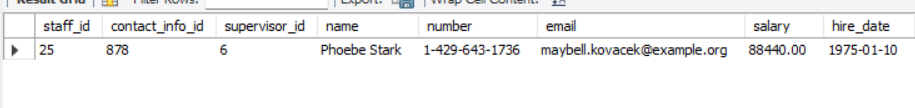


Рис. 7.5 – одна з трьох таблиць результату виконання процедури fetch\_agent\_related\_data. Таблиця з особистою інформацією

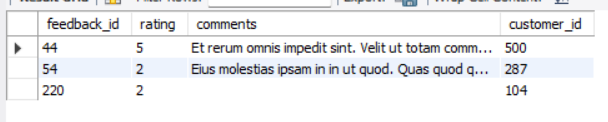


Рис. 7.6 – одна з трьох таблиць результату виконання процедури fetch\_agent\_related\_data. Таблиця з відгуками про агента

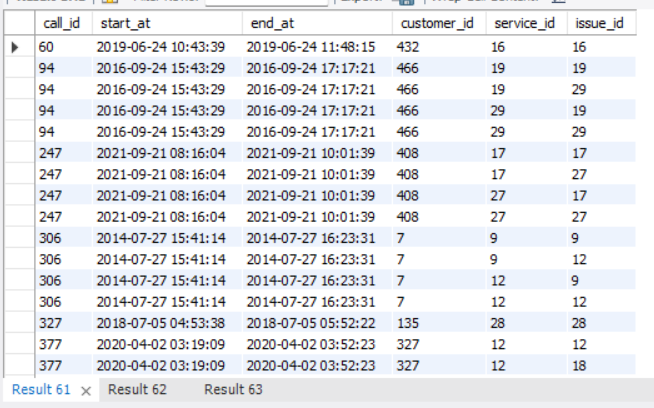


Рис. 7.7 – одна з трьох таблиць результату виконання процедури fetch\_agent\_related\_data. Детальна таблиця з проведеними дзвінками

7.2.6 Процедура increase\_salary\_for\_agent

Дана процедура створена для підвищення заробітної плати на конкретну суму конкретному агенту. Результат показан на рис. 7.8 та рис. 7.9

DELIMITER //

CREATE PROCEDURE increase\_salary\_for\_agent(IN agent\_id INT, IN special\_amount DECIMAL(7, 2))

BEGIN

DECLARE \_staff\_id INT;

DECLARE current\_salary DECIMAL(7, 2);

SELECT staff\_id INTO \_staff\_id FROM agent WHERE id = agent\_id;

SELECT salary INTO current\_salary FROM staff WHERE id = \_staff\_id;

IF current\_salary IS NOT NULL THEN

UPDATE staff SET salary = current\_salary + special\_amount WHERE id = \_staff\_id;

SELECT CONCAT('Salary increased by ', special\_amount, ' for Agent ID ', agent\_id) AS result;

ELSE

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Agent not found.';

END IF;

END //

DELIMITER ;

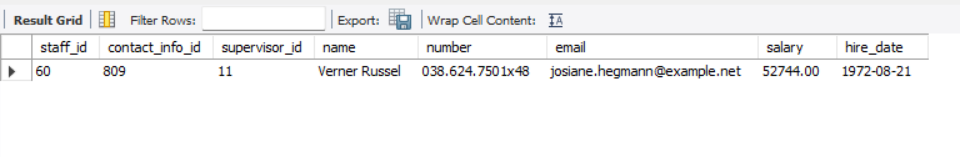


Рис. 7.8 – Зарплата агента до підвищення

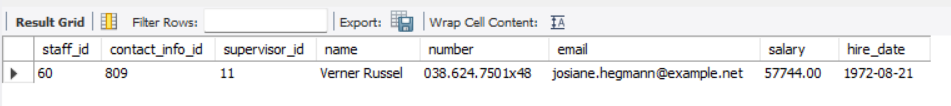


Рис. 7.9 – зарплата агента після підвищення.

7.2.7 Процедура get\_agent\_total\_earnings

Процедура створена для отримання загального заробітку з продаж послуг конкретного агента. Процедура може бути корисною для оцінки працьовитості агента та здібності продавати послуги. Результат для агента номер два на рис. 7.10.

DELIMITER //

CREATE PROCEDURE get\_agent\_total\_earnings(IN \_agent\_id INT)

BEGIN

DECLARE total\_earnings DECIMAL(10, 2);

SELECT s.salary + COALESCE(SUM(serv.price), 0) INTO total\_earnings

FROM staff s

LEFT JOIN agent a ON s.id = a.staff\_id

LEFT JOIN `call` c ON a.id = c.agent\_id

LEFT JOIN call\_service cs ON c.id = cs.call\_id

LEFT JOIN service serv ON cs.service\_id = serv.id

WHERE s.id = \_agent\_id;

IF total\_earnings IS NOT NULL THEN

SELECT CONCAT('Total earnings for Agent ID ', \_agent\_id, ': $', total\_earnings) AS result;

ELSE

SELECT CONCAT('No earnings available for Agent ID ', \_agent\_id) AS Result;

END IF;

END //

DELIMITER ;

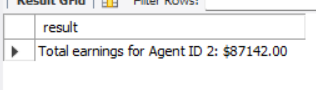


Рис. 7.10 – загальний заробіток агента з продаж послуг

7.2.8 Процедура update\_agent\_supervisor

Процедура створена для зручного оновлення наглядача для агента. Допомогає облегшити процес додавання та зміни даних. Результат оновлення процедури на рис. 7.11

DELIMITER //

CREATE PROCEDURE update\_agent\_supervisor(IN agentId INT, IN newSupervisorId INT)

BEGIN

IF EXISTS (SELECT 1 FROM agent WHERE id = agentId) AND EXISTS (SELECT 1 FROM supervisor WHERE id = newSupervisorId) THEN

UPDATE agent SET supervisor\_id = newSupervisorId WHERE id = agentId;

SELECT CONCAT('Supervisor updated for Agent ID ', agentId) AS Result;

ELSE

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Agent or supervisor not found.';

END IF;

END //

DELIMITER ;

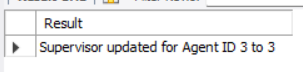


Рис. 7.11 – номер наглядача змінено для агента номер 3

7.2.9 Процедура average\_call\_duration

Дана процедура створена для отримання середньої тривалості розговору для кожного агента. Процедура допоможе дізнатися хто довше розмовляє з користувачами та більше їх цікавить. Результат процедури на рис. 7.12

DELIMITER //

CREATE PROCEDURE average\_call\_duration()

BEGIN

SELECT agent.id AS agent\_id, AVG(TIMESTAMPDIFF(SECOND, `call`.start\_at, `call`.end\_at)) AS average\_duration

FROM agent

JOIN `call` ON agent.id = `call`.agent\_id

GROUP BY agent.id

ORDER BY average\_duration;

END //

DELIMITER ;

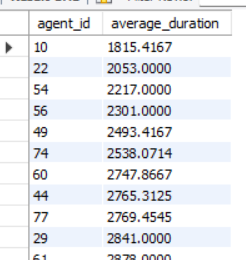


Рис. 7.12 – середня тривалість розмови кожного агента в секундах

7.2.10 Процедура most\_common\_issues

Процедура створена для виведення списку найчастіших предметів дискусії. Це допоможе зрозуміти найчастіши проблеми та зменшити їхню кількість або збільшити вартість послуг, які вирішують ці проблеми. Результат процедури на рис 7.13.

DELIMITER //

CREATE PROCEDURE most\_common\_issues()

BEGIN

SELECT `issue`.`id`, `issue`.`description`, COUNT(`call\_issue`.`call\_id`) AS `call\_count`

FROM `issue`

JOIN `call\_issue` ON `issue`.`id` = `call\_issue`.`issue\_id`

GROUP BY `issue`.`id`, `issue`.`description`

ORDER BY `call\_count` DESC;

END //

DELIMITER ;

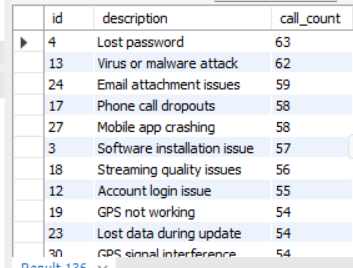


Рис. 7.13 – найчастіши предмети дискусії в розмові

7.3 Тригери

7.3.1 Тригер delete\_agent\_related\_data

Даний тригер створений для видалення відповідних записів з таблиць, які були пов’язані з агентом. Для виведення результати видалення даних для користувача під номером один використаємо процедуру fetch\_agent\_related\_data на рис. 7.14, рис. 7.15 та рис. 7.16.

DELIMITER //

CREATE TRIGGER delete\_agent\_related\_data

AFTER DELETE ON agent

FOR EACH ROW

BEGIN

DECLARE \_staff\_id INT;

DECLARE \_contact\_info\_id INT;

SELECT staff\_id INTO \_staff\_id FROM agent WHERE id = OLD.id;

SELECT contact\_info\_id INTO \_contact\_info\_id FROM staff WHERE id = \_staff\_id;

DELETE FROM call\_service WHERE call\_id IN (SELECT id FROM `call` WHERE agent\_id = OLD.id);

DELETE FROM call\_issue WHERE call\_id IN (SELECT id FROM `call` WHERE agent\_id = OLD.id);

DELETE FROM `call` WHERE agent\_id = OLD.id;

DELETE FROM feedback WHERE agent\_id = OLD.id;

DELETE FROM staff WHERE id = \_staff\_id;

DELETE FROM contact\_info WHERE id = \_contact\_info\_id;

END //

DELIMITER ;

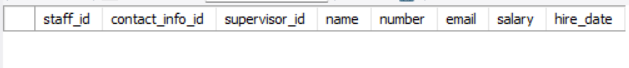


Рис. 7.14 – Уся особиста інформація про агента видалена тригером.

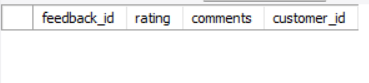


Рис. 7.15 – Усі відгуки про агента видалені тригером.

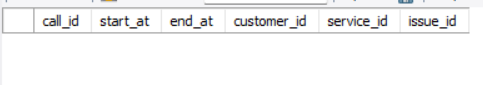


Рис. 7.16 – Усі дані про дзвінки агента видалено

7.3.2 Тригер delete\_supervisor\_related\_data

Даний тригер аналогічен минулому – створений для видалення відповідних даних, які належали наглядачу. Також, він призначає агентам нового наглядача, в якого найменша кількість агентів у підпорядкуванні. Результати видалення наглядача під номером 1 на рис. 7.17 та рис. 7.18.

DELIMITER //

CREATE TRIGGER delete\_supervisor\_related\_data

AFTER DELETE ON supervisor

FOR EACH ROW

BEGIN

DECLARE \_staff\_id INT;

DECLARE \_contact\_info\_id INT;

DECLARE \_new\_supervisor\_id INT;

SELECT staff\_id INTO \_staff\_id FROM supervisor WHERE id = OLD.id;

SELECT contact\_info\_id INTO \_contact\_info\_id FROM staff WHERE id = \_staff\_id;

-- Пошук наглядача з найменшою кількістю агентів

SELECT id INTO \_new\_supervisor\_id FROM supervisor

WHERE id != OLD.id

ORDER BY (SELECT COUNT(\*) FROM agent WHERE supervisor\_id = supervisor.id) ASC

LIMIT 1;

-- Перерозподіл агентів до нового наглядача

UPDATE agent SET supervisor\_id = \_new\_supervisor\_id WHERE supervisor\_id = OLD.id;

DELETE FROM staff WHERE id = \_staff\_id;

DELETE FROM contact\_info WHERE id = \_contact\_info\_id;

END //

DELIMITER ;

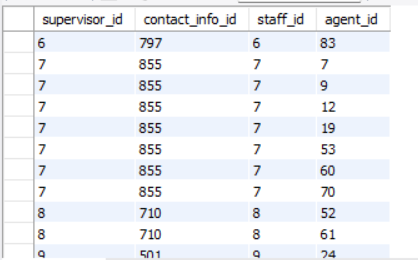


Рис. 7.17 – Дані наглядачів та їхніх агентів до видалення

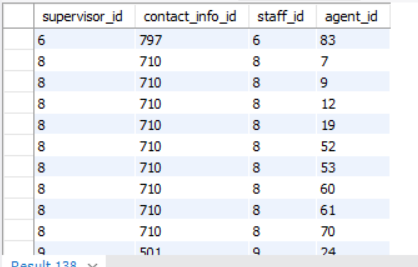


Рис. 7.18 – Дані наглядачів після видалення наглядача під номером 7. Усі підпорядковані йому агенти перейшли до наглядача 8

7.3.3 Тригер delete\_customer\_related\_record

Даний тригер схожий на два минулих – він створений для видалення відповідних записів з таблиці contact\_info, які належали користувачу. Результати видалення всіх даних користувача номер 21 на прикладі його дзвінків рис. 7.19 та рис. 7.20.

DELIMITER //

CREATE TRIGGER delete\_customer\_related\_data

AFTER DELETE ON customer

FOR EACH ROW

BEGIN

DELETE FROM call\_service WHERE call\_id IN (SELECT id FROM `call` WHERE customer\_id = OLD.id);

DELETE FROM call\_issue WHERE call\_id IN (SELECT id FROM `call` WHERE customer\_id = OLD.id);

DELETE FROM `call` WHERE customer\_id = OLD.id;

DELETE FROM feedback WHERE customer\_id = OLD.id;

DELETE FROM contact\_info WHERE id = OLD.contact\_info\_id;

END //

DELIMITER ;

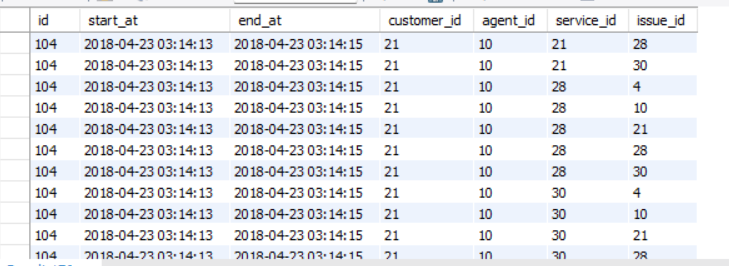


Рис. 7.19 – Дані дзвінків користувача номер 21 до видалення

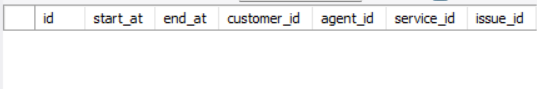


Рис. 7.20 – Дані дзвінків користувача номер 21 після видалення

7.3.4 Тригер delete\_call\_related\_record

Даний тригер створений для видалення записів дзвінків та відповідних записів з таблиць предметів дискусії та послуг. Результати виконання на прикладі дзінка номер 10 на рис. 7.21 та 7.2

DELIMITER //

CREATE TRIGGER delete\_call\_related\_data

AFTER DELETE ON `call`

FOR EACH ROW

BEGIN

DELETE FROM call\_service WHERE call\_id = OLD.id;

DELETE FROM call\_issue WHERE call\_id = OLD.id;

END //

DELIMITER ;

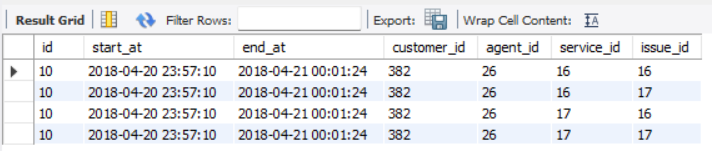


Рис. 7.21 – Дані дзвінків номер 10 до видалення

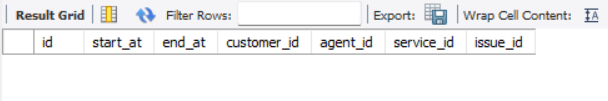


Рис. 7.22 – Дані дзвінків номер 10 після видалення

7.3.5 Тригер auto\_assign\_supervisor

Даний тригер створений для автоматичного назначення наглядача за агентом, в залежності від кількості агентів у підпорядкувані наглядачу. Результат роботи на новому працівнику під номером 101 на рис. 7.23.

DELIMITER //

CREATE TRIGGER auto\_assign\_supervisor

BEFORE INSERT ON agent

FOR EACH ROW

BEGIN

DECLARE \_new\_supervisor\_id INT;

SELECT id INTO \_new\_supervisor\_id FROM supervisor

ORDER BY (SELECT COUNT(\*) FROM agent WHERE supervisor\_id = supervisor.id) ASC

LIMIT 1;

SET NEW.supervisor\_id = \_new\_supervisor\_id;

END //

DELIMITER ;

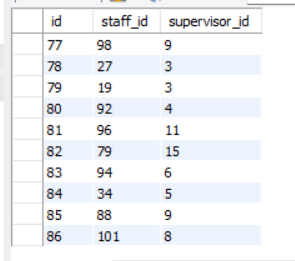


Рис. 7.23 – Агент під номером 86 має наглядача номер 8

7.4 Представлення

7.4.1 Представлення StaffView

Представлення StaffView містить повну інформацію про всіх співробітників call-центру. Вигляд представлення на рис. 7.24.

CREATE VIEW StaffView AS

SELECT

s.id AS staff\_id,

s.salary,

s.hire\_date,

c.name AS staff\_name,

c.number AS staff\_contact\_number,

sv.id AS supervisor\_id

FROM staff s

JOIN contact\_info c ON s.contact\_info\_id = c.id

LEFT JOIN supervisor sv ON s.id = sv.staff\_id;

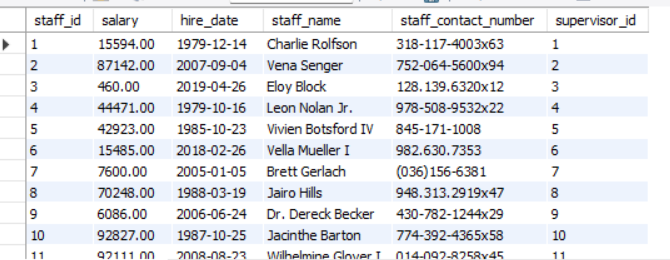


Рис. 7.24 – Вигляд представлення StaffView

7.4.2 Представлення CallDetailsView

Представлення CallDetailsView містить інформацію повну інформацію про дзвінки, назву послуги, предмет дискусії. Вигляд представлення на рис. 7.25.

CREATE VIEW CallDetailsView AS

SELECT

ca.id AS call\_id,

ca.start\_at,

ca.end\_at,

cu.id AS customer\_id,

a.id AS agent\_id,

s.name AS service\_name,

iss.description AS issue\_description

FROM `call` ca

JOIN customer cu ON ca.customer\_id = cu.id

JOIN agent a ON ca.agent\_id = a.id

LEFT JOIN call\_service cs ON ca.id = cs.call\_id

LEFT JOIN service s ON cs.service\_id = s.id

LEFT JOIN call\_issue ci ON ca.id = ci.call\_id

LEFT JOIN issue iss ON ci.issue\_id = iss.id;

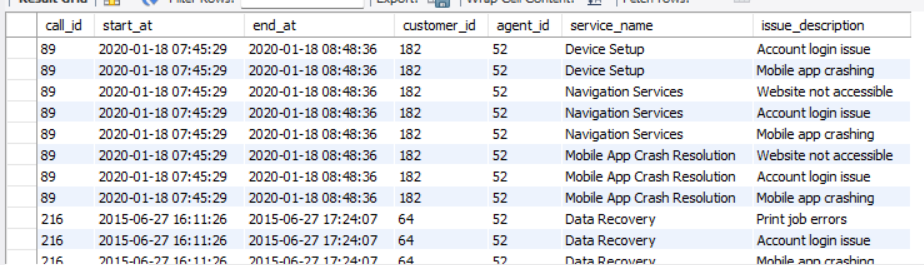


Рис. 7.25 – Вигляд представлення CallDetailsView

7.4.3 Представлення CustomerFeedbackView

Представлення CustomerFeedbackView створено для відображення інформації про користувача та відгук, що він залишив. Представлення необхідно для оцінки загальної картини серед роботи агентів. Вигляд представлення на рис. 7.26.

CREATE VIEW CustomerFeedbackView AS

SELECT

c.id AS customer\_id,

ci.`name` AS customer\_name,

ci.`number` AS customer\_contact\_number,

a.id AS agent\_id,

f.id AS feedback\_id,

f.rating,

f.comments

FROM

feedback f

JOIN customer c ON f.customer\_id = c.id

JOIN agent a ON f.agent\_id = a.id

JOIN contact\_info ci ON c.contact\_info\_id = ci.id;

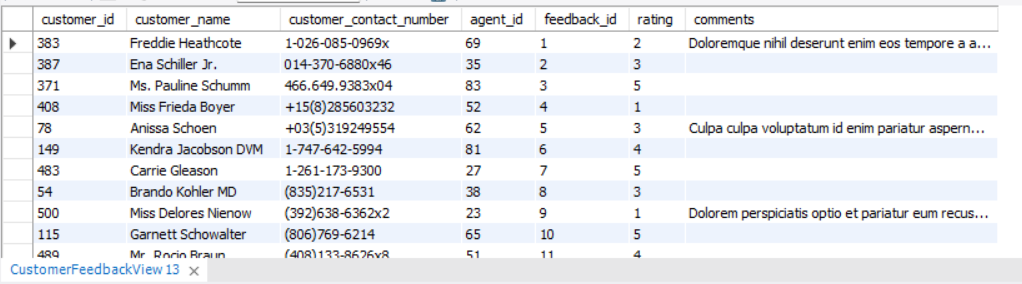


Рис. 7.26 – Вигляд представлення CustomerFeedbackView

7.5 SQL-запити

1) Запит на отримання кількості клієнтів, що використовують кожен сервіс(результат на рис. 7.27):

SELECT s.id AS service\_id, s.name, COUNT(cs.call\_id) AS usage\_count

FROM service s

LEFT JOIN call\_service cs ON s.id = cs.service\_id

GROUP BY s.id;

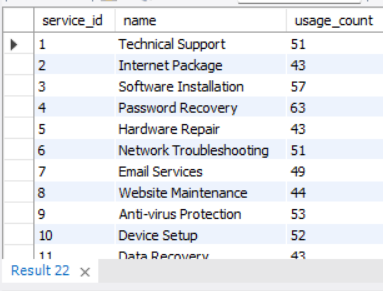


Рис. 7.27 – Результат виконання запиту 1

2) Запит на отримання п’яти агентів з найбільшою кількістю вігдуків (результат на рис. 7.28):

SELECT a.id AS agent\_id, COUNT(f.id) AS feedback\_count

FROM agent a

LEFT JOIN feedback f ON a.id = f.agent\_id

GROUP BY a.id

ORDER BY feedback\_count DESC

LIMIT 5;

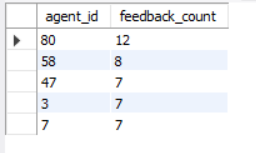


Рис. 7.28 – Результат виконання запиту 2

3) Запит на отримання п’яти послуг, з найкращим середнім рейтингом(результат на рис. 7.29):

SELECT s.id AS service\_id, s.name, AVG(s.price) AS avg\_price

FROM service s

GROUP BY s.id

ORDER BY avg\_price DESC

LIMIT 5;

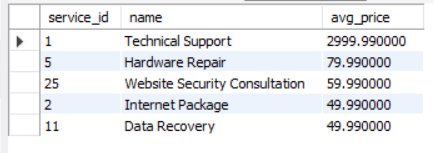


Рис. 7.29 – Результат виконання запиту 3

4) Запит для отримання кількості агентів, які не отримали відгук (результат на рис. 7.30):

SELECT COUNT(\*) AS agents\_without\_feedback

FROM agent a

LEFT JOIN feedback f ON a.id = f.agent\_id

WHERE f.id IS NULL;

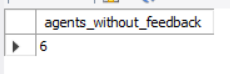


Рис. 7.30 – Результат виконання запиту 4

5) Запит для отримання проблеми, яка найчастіше виникає в звітах про виклики (результат на рис. 7.31):

SELECT i.id AS issue\_id, i.description AS common\_issue, COUNT(ci.issue\_id) AS occurrence\_count

FROM issue i

LEFT JOIN call\_issue ci ON i.id = ci.issue\_id

GROUP BY i.id

ORDER BY occurrence\_count DESC

LIMIT 1;

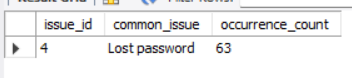


Рис. 7.31 – Результат виконання запиту 5

6) Запит для обчислення рейтингу для кожної послуги на основі розмов користувачів та відгуків, що вони залишили (результат на рис. 7.32):

SELECT s.id AS service\_id, s.name AS service\_name, AVG(f.rating) AS average\_rating

FROM service s

JOIN call\_service cs ON s.id = cs.service\_id

JOIN `call` c ON cs.call\_id = c.id

JOIN feedback f ON c.customer\_id = f.customer\_id

GROUP BY s.id;

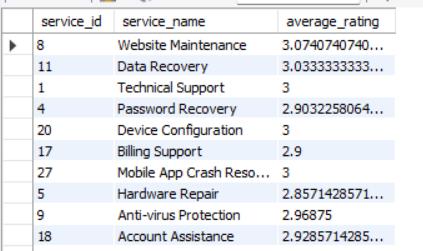


Рис. 7.32 – Результат виконання запиту 6

7) Запит розраховує середню кількість питань на один виклик (результат на рис. 7.33):

SELECT COUNT(ci.issue\_id) / COUNT(DISTINCT ca.id) AS avg\_issues\_per\_call

FROM `call` ca

LEFT JOIN call\_issue ci ON ca.id = ci.call\_id;

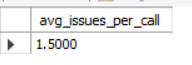


Рис. 7.33 – Результат виконання запиту 7

8) Запит показує кількість викликів для кожної години у добі (результат на рис. 7.34):

SELECT HOUR(start\_at) AS hour\_of\_day, COUNT(id) AS call\_count

FROM `call`

GROUP BY hour\_of\_day;

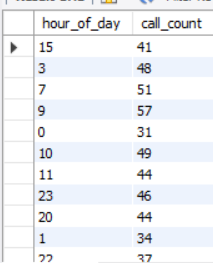


Рис. 7.34 – Результат виконання запиту 8

9) запит визначає трьох агентів з найбільшою кількістю викликів, в яких були предмети дискусії. (результат на рис. 7.35):

SELECT a.id AS agent\_id, COUNT(ci.call\_id) AS issue\_call\_count

FROM agent a

LEFT JOIN call\_issue ci ON a.id = ci.issue\_id

GROUP BY a.id

ORDER BY issue\_call\_count DESC

LIMIT 3;

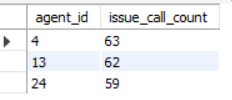


Рис. 7.35 – Результат виконання запиту 9

10) Запит рахує кількість викликів для кожної послуги кожного місяця (результат на рис. 7.36):

)

SELECT s.name AS service\_name, MONTH(ca.start\_at) AS month, COUNT(cs.call\_id) AS call\_count

FROM service s

LEFT JOIN call\_service cs ON s.id = cs.service\_id

LEFT JOIN `call` ca ON cs.call\_id = ca.id

GROUP BY s.name, month;

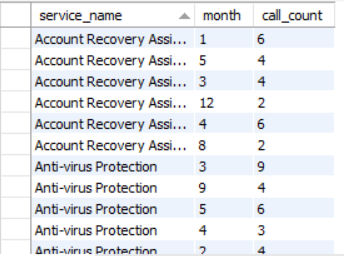


Рис. 7.36 – Результат виконання запиту 10

11) Запит рахує кількість агентів, які працювали у вихідні дні (результат на рис. 7.37):

SELECT COUNT(DISTINCT a.id) AS weekend\_agent\_count

FROM agent a

JOIN `call` ca ON a.id = ca.agent\_id

WHERE DAYOFWEEK(ca.start\_at) IN (1, 7);



Рис. 7.37 – Результат виконання запиту 11

12) Запит визначає кількість агентів, які не отримали жодного відгуку для відповідної наданої послуги (результат на рис. 7.38):

SELECT s.id AS service\_id, COUNT(DISTINCT a.id) AS agents\_without\_feedback\_count

FROM service s

LEFT JOIN call\_service cs ON s.id = cs.service\_id

LEFT JOIN `call` ca ON cs.call\_id = ca.id

LEFT JOIN agent a ON ca.agent\_id = a.id

LEFT JOIN feedback f ON a.id = f.agent\_id

WHERE f.id IS NULL

GROUP BY s.id;



Рис. 7.38 – Результат виконання запиту 12

13) Запит визначає загальну тривалість розмов для кожного агента (результат на рис. 7.39):

SELECT

a.id AS agent\_id,

ci.name AS agent\_name,

SUM(TIMESTAMPDIFF(SECOND, c.start\_at, c.end\_at)) AS total\_duration\_seconds

FROM agent a

JOIN staff s ON a.staff\_id = s.id

JOIN contact\_info ci ON s.contact\_info\_id = ci.id

JOIN `call` c ON a.id = c.agent\_id

GROUP BY a.id, ci.name;;

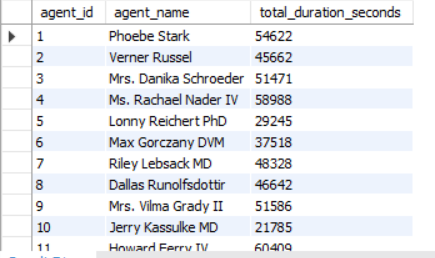


Рис. 7.39 – Результат виконання запиту 13

14) Запит для отримання списку клієнтів, що не залишили відгука (результат на рис. 7.40):

SELECT cu.id AS customer\_id, ci.`name` AS customer\_name

FROM customer cu

JOIN contact\_info ci ON cu.contact\_info\_id = ci.id

WHERE cu.id NOT IN (SELECT DISTINCT customer\_id FROM feedback);

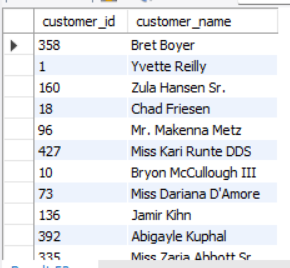


Рис. 7.40 – Результат виконання запиту 14

15) Запит для кількості заробітку кожної послуги (результат на рис. 7.41):

SELECT s.name AS service\_name, SUM(s.price) AS total\_earnings

FROM call\_service cs

JOIN service s ON cs.service\_id = s.id

GROUP BY s.name;

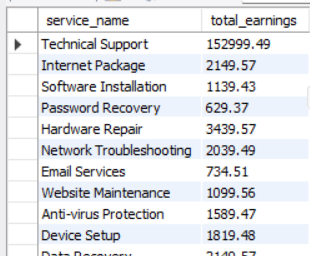


Рис. 7.41 – Результат виконання запиту 15

16) Запит для отримання спіску користувачів, які дали найвищу оцінку (результат на рис. 7.42):

SELECT ci.name AS customer\_name, f.rating AS highest\_rating

FROM customer cu

JOIN contact\_info ci ON cu.contact\_info\_id = ci.id

JOIN feedback f ON cu.id = f.customer\_id

WHERE f.rating = (SELECT MAX(rating) FROM feedback);



Рис. 7.42 – Результат виконання запиту 16

17) Запит для отримання номерів агента, який мав найбільшу кількість дзвінків у травні (результат на рис. 7.43):

SELECT

a.id AS agent\_id,

a.staff\_id,

COUNT(c.id) AS call\_count

FROM agent a

JOIN `call` c ON a.id = c.agent\_id

WHERE MONTH(c.start\_at) = 5 -- Specify the desired month

GROUP BY a.id, a.staff\_id

ORDER BY call\_count DESC

LIMIT 1;

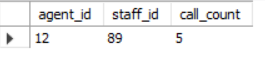


Рис. 7.43 – Результат виконання запиту 17

18) Запит для отримання списку розмов з ціною послуг більше ніж середня (результат на рис. 7.44):

SELECT

ca.id AS call\_id,

ca.start\_at,

ca.end\_at,

s.name AS service\_name,

s.price

FROM `call` ca

JOIN call\_service cs ON ca.id = cs.call\_id

JOIN service s ON cs.service\_id = s.id

WHERE s.price > (SELECT AVG(price) FROM service);

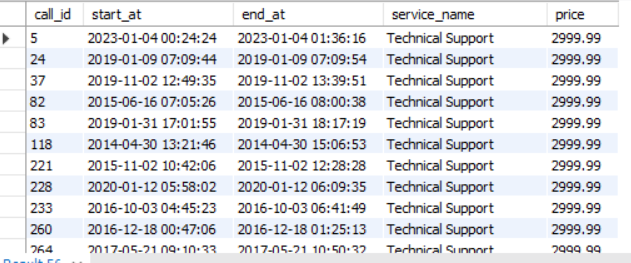


Рис. 7.44 – Результат виконання запиту 18

19) Запит для визначення агента з найвищим середнім рейтингом враховуючи тільки агентів у яких хоча б 2 відгуки (результат на рис. 7.45):

SELECT

a.id AS agent\_id,

a.staff\_id,

AVG(CAST(f.rating AS DECIMAL(3, 2))) AS average\_rating

FROM agent a

LEFT JOIN feedback f ON a.id = f.agent\_id

WHERE a.id IN (

SELECT agent\_id

FROM feedback

GROUP BY agent\_id

HAVING COUNT(\*) >= 2

)

GROUP BY a.id, a.staff\_id

ORDER BY average\_rating DESC

LIMIT 1;

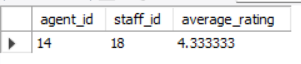


Рис. 7.45 – Результат виконання запиту 19

20) Запит для отримання агенту з найбільшою кількістю дзвінків. Відображає інформацію про агента та середню тривалість розмови (результат на рис. 7.46):

SELECT

a.id AS agent\_id,

a.staff\_id,

COUNT(c.id) AS total\_calls,

AVG(TIMESTAMPDIFF(SECOND, c.start\_at, c.end\_at)) AS average\_call\_duration

FROM agent a

JOIN `call` c ON a.id = c.agent\_id

WHERE a.id = (

SELECT agent\_id

FROM `call`

GROUP BY agent\_id

ORDER BY COUNT(id) DESC

LIMIT 1

)

GROUP BY a.id, a.staff\_id;

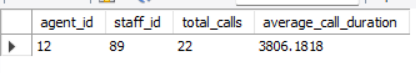


Рис. 7.46 – Результат виконання запиту 20

7.6 Індекси та результати оптимізації

Для оптимізації роботи бази даних були створені індекси для даних в таблицях для швидкого пошуку.

Запит 1: Середній рейтинг для кожної послуги

SELECT s.id AS service\_id, s.name AS service\_name, AVG(f.rating) AS average\_rating

FROM service s

JOIN call\_service cs ON s.id = cs.service\_id

JOIN `call` c ON cs.call\_id = c.id

JOIN feedback f ON c.customer\_id = f.customer\_id

GROUP BY s.id;

Створення індексів для оптимізації:

CREATE INDEX idx\_service\_id ON call\_service (service\_id);

CREATE INDEX idx\_call\_id ON call\_service (call\_id);

CREATE INDEX idx\_customer\_id ON `call` (customer\_id);

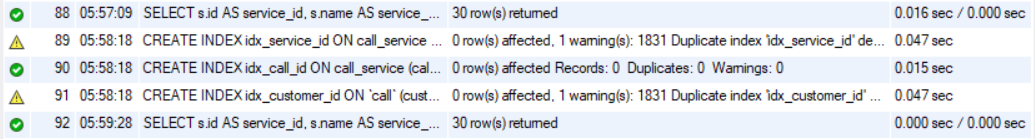


Рис. 7.47 – Час виконання запиту 1 до (0.016 секунд), та після оптимізації (0.000 секунд)

Запит 2: Сумарна тривалість дзвінків для кожного агента

SELECT

a.id AS agent\_id,

ci.name AS agent\_name,

SUM(TIMESTAMPDIFF(SECOND, c.start\_at, c.end\_at)) AS total\_duration\_seconds

FROM agent a

JOIN staff s ON a.staff\_id = s.id

JOIN contact\_info ci ON s.contact\_info\_id = ci.id

JOIN `call` c ON a.id = c.agent\_id

GROUP BY a.id, ci.name;

Створення індексів для оптимізації:

CREATE INDEX idx\_agent\_id ON `call` (agent\_id);

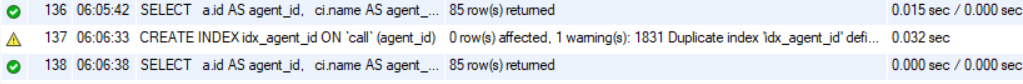


Рис. 7.48 – Час виконання запиту 2 до (0.015 секунд), та після оптимізації (0.000 секунд)

Запит 3: Найпопулярніша проблема серед відгуків

SELECT i.id AS issue\_id, i.description AS popular\_issue, COUNT(ci.issue\_id) AS occurrence\_count

FROM issue i

LEFT JOIN call\_issue ci ON i.id = ci.issue\_id

GROUP BY i.id

ORDER BY occurrence\_count DESC

LIMIT 1;

Створення індексів для оптимізації:

CREATE INDEX idx\_issue\_id ON call\_issue (issue\_id);

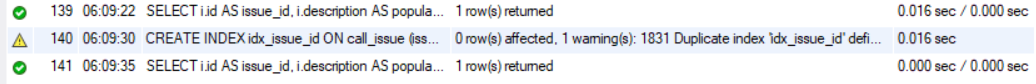


Рис. 7.49 – Час виконання запиту 3 до (0.016 секунд), та після оптимізації (0.000 секунд)

Можна впевнено зазначити, що вдалося оптимізувати роботи бази даних за допомогою створення індексів для полей у таблицях. Час виконання запитів зменшився в середньому на 0.015 секунд.

Під час роботи над цим розділом було створено процедури, представлення та виконано велику кількість SQL запитів до бази даних call-центру.

**Висновки**

Під час виконання курсової роботи я провів аналіз предметного середовища та встановив чіткі вимоги для створення бази даних для підтримки діяльності call-центру. Протягом аналізу матеріалу, за допомогою існуючих програмних продуктів, встановив бізнес правила та розробив ER-модель бази даних.

Розроблена ER-модель відображає всі наявні сутності та зв’язки між ними. Сама бази даних включає створення таблиць, імпортування даних, створення багатокористувацького інтерфейсу, а також розробку генераторів, збережених процедур, тригерів, представлень та SQL-запитів.

У підсумку була створена ефективна та найдіна база даних для кол-центру, що забезпечить автоматизацію поширених процесів, а також надасть доступну звітність про успішність роботи агентів.

**Список використаної літератури**

1. СУБД MySQL. URL: <https://www.mysql.com/> (дата звернення: 20.01.2024).
2. Хмарні рішення для кол-центрів Five9. URL: <https://www.five9.com/products/capabilities/global-voice> (дата звернення: 20.01.2024).
3. Програмне забезпечення для кол-центрів VoIPTime Contact Center. URL: <https://www.voiptime.net/uk/> (дата звернення: 20.01.2024).
4. The Open source definition. URL: <https://opensource.org/osd/> (last accessed: 20.01.2024).
5. MySQL Workbench. URL: <https://www.mysql.com/products/workbench/> (last accessed: 20.01.2024).
6. Сайт для генерації даних. URL: <https://cobbl.io/> (дата звернення: 21.01.2024).
7. Сайт для генерації даних. URL: <https://filldb.info/> (дата звернення 21.01.2024).